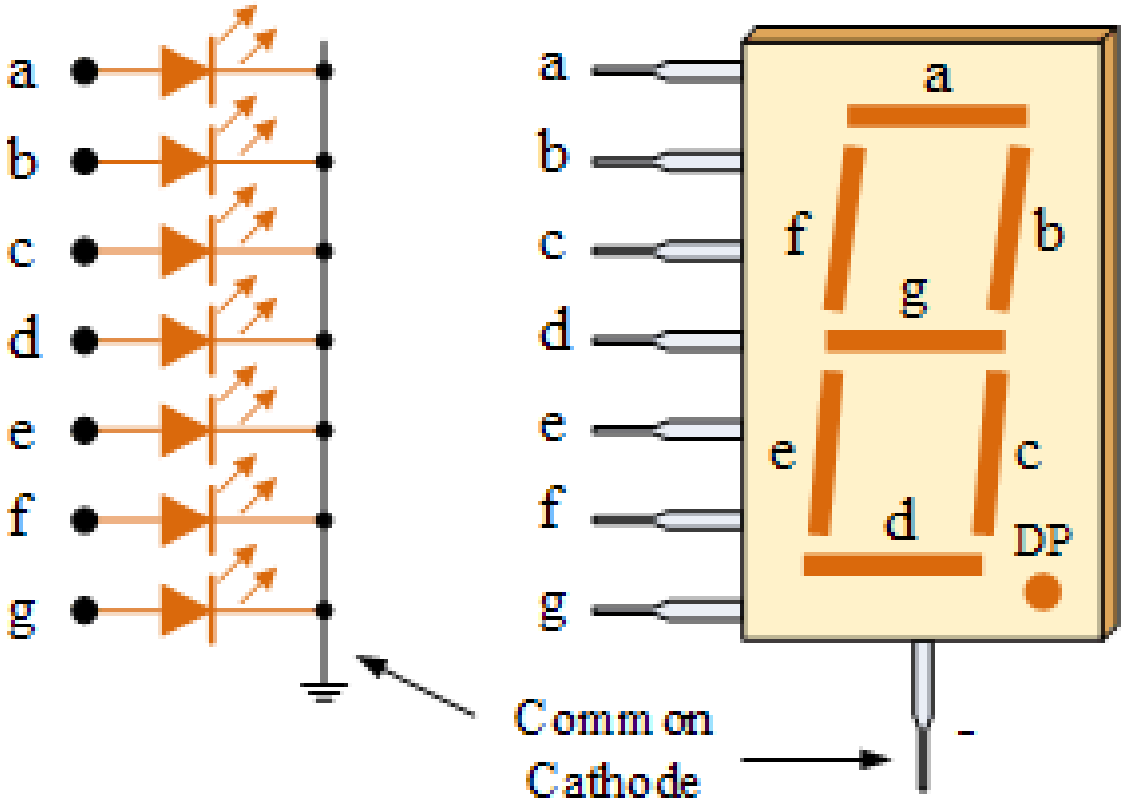
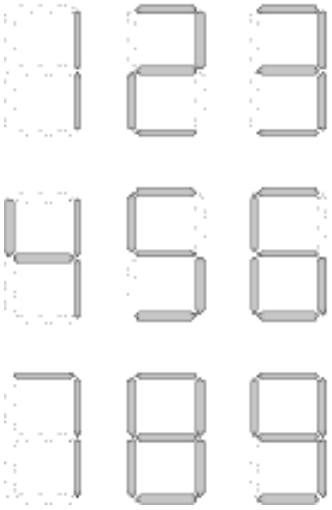
Ortak katotlu 7 segmentli bir dijital gösterge kullanarak sürekli olacak şekilde 0 dan 9'a kadar sayacak bir devre kurup programlayınız.





Digit Shown	Illuminated Segment (1 = illumination)						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	0	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Kod:

```
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = ON
#pragma config IESO = OFF
#pragma config FCMEN = OFF

#include <xc.h>
#define _XTAL_FREQ 4000000

unsigned char const SEGMENT_MAP[10] = {0x3F, 0x06, 0x5B,
0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

void main(void) {
    TRISC = 0x00;
    char digit = 0;

    while (1) {
        PORTC = (SEGMENT_MAP[digit]);
        __delay_ms(200);
        digit++;
        if (digit > 9)digit = 0;
        else {
            PORTB = (SEGMENT_MAP[digit]);
        }
    }
}
```

Alternatif Kod:

```
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = ON
#pragma config IESO = OFF
#pragma config FCMEN = OFF

#include <xc.h>
#define _XTAL_FREQ 4000000

void main(void) {
    TRISC = 0x00;

    while (1) {
        PORTC = 0x3F;
        __delay_ms(200);
        PORTC = 0x06;
        __delay_ms(200);
        PORTC = 0x5B;
        __delay_ms(200);
        PORTC = 0x4F;
        __delay_ms(200);
        PORTC = 0x66;
        __delay_ms(200);
        PORTC = 0x6D;
        __delay_ms(200);
        PORTC = 0x7D;
        __delay_ms(200);
        PORTC = 0x07;
        __delay_ms(200);
        PORTC = 0x7F;
        __delay_ms(200);
        PORTC = 0x6F;
        __delay_ms(200);
    }
}
```

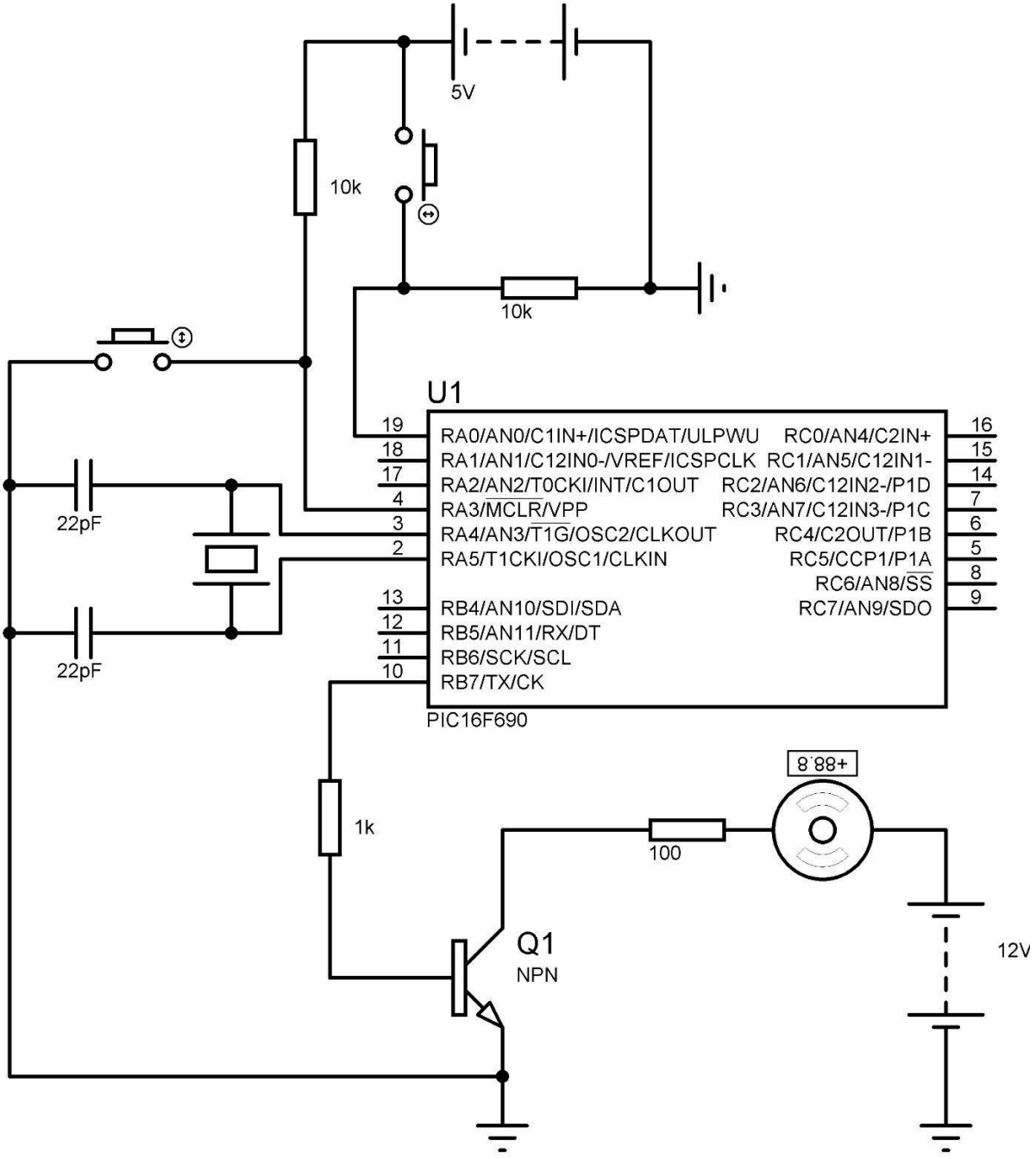
Açıklama:

Algoritmayı nasıl oluşturduğunuza bağlı olarak bir çok farklı şekilde çözülebilir. Yukarıdaki her iki örnekte de dahili osilatörün varsayılan (default) değeri kullanılarak işlem yapılmıştır. Bu da bizim harici ekstradan osilatör bağlama gereksinimimizi ortadan kaldırmaktadır. Resetleme ucu da deaktif edilerek Master Clear ucunun enerjilendirilmesi gereksinimi de ortadan kaldırılmıştır. PORTC'nin tamamı da (0'dan 7'ye kadar olan pinleri) çıkış yapılarak bu işlem gerçekleştirilmiştir.

Burada ekstradan dikkat edilmesi gereken şey, önceden bir dizi tanımlanarak ortak katotlu göstergemizi haritalandırmak. Ortak katot demek, söz konusu LED'lerin tamamının toprak ucunun bir noktada olması demek. Bu durumda herhangi bir LED'i yakmak demek ilgili uca lojik 1 bilgisi göndermek demektir. Buna uygun şekilde hangirakama hangi kombinasyonun denk geleceğini tablo oluşturarak karar veririz.

Ortak anotlu denilirse eğer, yukarıda bahsettiğimiz durumun tam tersi bir olay söz konusu, yani, LED'lerin enerji uçları ortak toprak uçları bağımsızdır. Bu durumda da hangi LED yakılmak isteniyorsa lojik 0 bilgisi gönderilmeli. Aşağıda ortak katotlu bir göstergeye ait bağlantılar ve bu bağlantılara gönderilmesi gereken bilgiyi içeren tablogösterilmiştir.

Proteusta DC motoru tek yönlü olarak çalıştıracak şekilde aşağıda verilen devreyi oluşturunuz. Bu devrede belirlediğiniz bir pine lojik 1 bilgisi verildiğinde motor çalışacak, lojik 0 bilgisi verildiğinde de motor duracaktır.



Kod:

```
#pragma config FOSC
= EXTRCIO#pragma
config WDTE = OFF
#pragma config PWRTE
= OFF #pragma config
MCLRE = ON #pragma
config CP = OFF
#pragma config CPD =
OFF #pragma config
BOREN = ON #pragma
config IESO = OFF
#pragma config FCMEN
= OFF #define
_XTAL_FREQ 8000000
#include <xc.h>

    int main() {
ANSEL = 0x00;
ANSELH = 0x00;
TRISAbits.TRISA0 = 1;
TRISBbits.TRISB7 = 0;
PORTBbits.RB7 = 0;
while (1) {
    if (PORTAbits.RA0 == 1) {
        PORTBbits.RB7 = 1;
        __delay_ms(2);
    } else {
        PORTBbits.RB7 = 0;
        __delay_ms(2);
    }

}
    }
```

Açıklama:

Hangi pinleri giriş ve hangilerini de çıkış yapacağınıza TRIS registerları ile karar veriyorsunuz. Tabi dijital bilgi alıp, dijital bilgi göndereceğimiz için ANSEL ve ANSELH registerlarını sıfırlıyoruz başlarken. Yukarıdaki örnekte 8 MHz değerinde harici osilatörkullanarak sistem saati belirlenmiştir. Bu da bizim harici ekstradan osilatör bağlamamız gerektiğini belirtmektedir. Resetleme ucunu da aktif ederek Master Clearucunun enerjilendirilmesi gereksinimini meydana getirdik. PORTA'nın 0 nolu pinini giriş, PORTB'nin 7 nolu pinini de çıkış olarak ayarladık.

Mikrodenetleyicimiz maximum 25 mA çıkış akımı sağlayabildiğinden bu bizim motoru sürmemize yetmeyecek ve direk sürmek için bağlamamız durumunda da entegremize hasar verecektir. Bu sebeple bir adet transistör kullanarak bu problemi çözmüş oluyoruz (Devreden de görüleceği üzere). Transistöre Lojik 1 bilgisi gönderip iletime sokmakta ve motorun ihtiyaç duyacağı yüksek akımı bu transistör aracılığıyla ayrı bir kaynak kullanarak karşılamaktayız. Lojik 1'de transistör iletimde olacağı için akım akacak ve motor dönecek, lojik 0'da ise transistör kesimde olacak ve akım akmayacağıiçin motor duracaktır.